1. Public int countLeftleaves(BSTNode curr) {

If (curr == null) {

Return 0;

}

If (curr.Left != null && curr.left.left == null && curr.left.right == null) {

Return 1 + countLeftleaves(curr.right);

}

Return countLeftLeaves(curr.left) + countLeftLeaves(curr.right);

}

1. Public void removeAllLeaves() {

Root = removeAllLeaves(root);

}

Private BSTNode removeAllLeaves(BSTNode curr) {

If (curr == null) {

Return null;

}

If (curr.left == null and curr.right == null) {

Return null;

}

Curr.left = removeAllLeaves(curr.left);

Curr.right = removeAllLeaves(curr.right);

Return curr;

}

1. Public int sumRightNodes() {

Return sumRightNodes(root);

}

Private int sumRightNodes(BSTNode curr) {

If (curr == null) {

Return 0;

}

If (curr.right != null) {

Return curr.right.data + sumRightNodes(curr.left) + sumRightNodes(curr.right);

}

Return sumRightNodes(curr.left) + sumRightNodes(curr.right);

}

1. Public int countGaps(Node curr) {

If (curr == null) {

Return 0;

}

If (curr. Left == null and curr.right == null) {

Return 2;

} else if (curr. Left != null and curr.right == null) {

Return 1 + countgaps(curr.left);

} else if (curr. Left == null and curr.right != null) {

Return 1 + countGaps(curr.right);

} else {

Return countGaps(curr.left) + countGaps(curr.right);

}

}

1. Public int countLessThanX(int x, Node curr) {

If (curr == null) {

Return 0;

}

If (curr.data < x) {

Return 1 + countLessThanX(curr.left) + countLessThanX(curr.right);

}

Return countLessThanX(curr.left);

}

1. Public int sumLessThanX(int x, Node curr) {

If (curr == null) {

Return 0;

}

If (curr.data < x) {

Return curr.data + sumLessThanX(curr.left) + sumLessThanX(curr.right);

}

Return sumLessThanX(curr.left);

}

1. Public int sumOddNumbers() {

Return sumOddNumber(root);

}

Private int sumOddNumbers(Node curr) {

If (curr == null) {

Return 0;

}

If (curr.data % 2 != 0) {

Return curr.data + sumOddNodes(curr.left) + sumOddNodes(curr.right);

}

Return sumOddNodes(curr.left) + sumOddNodes(curr.right);

}

1. Invert a binary tree

public void reverseRecursive(TreeNode treeNode) {

if(treeNode == null) {

return;

}

TreeNode temp = treeNode.getLeftChild();

treeNode.setLeftChild(treeNode.getRightChild());

treeNode.setRightChild(temp);

reverseRecursive(treeNode.getLeftChild());

reverseRecursive(treeNode.getRightChild());

}